On the Characteristics of Sequential Decision Problems and Their Impact on Evolutionary Computation and Reinforcement Learning

André M. S. Barreto[1], Douglas A. Augusto[2], and Helio J. C. Barbosa[1]

[1] Laboratório Nacional de Computação Científica
Petrópolis, RJ, Brazil
{amsb,hcbm}@lncc.br
[2] COPPE–Universidade Federal do Rio de Janeiro,
Rio de Janeiro, RJ, Brazil
douglas@coc.ufrj.br

Abstract. This work provides a systematic review of the criteria most commonly used to classify sequential decision problems and discusses their impact on the performance of reinforcement learning and evolutionary computation. The paper also proposes a further division of one class of decision problems into two subcategories, which delimits a set of decision tasks particularly difficult for optimization techniques in general and evolutionary methods in particular. A simple computational experiment is presented to illustrate the subject.

1 Introduction

In a sequential decision-making problem the consequences of a decision may last for an arbitrarily long time [13]. Thus, a choice that seems beneficial from a short-sighted perspective may reveal itself to be disastrous in the long run. In the game of chess, for example, a move that captures one of the opponent's pieces may also expose the king and eventually lead to a defeat.

Many real-world tasks involve this tradeoff between immediate and long-term benefits. Problems of practical and economical interest arising in areas as diverse as operations research, control theory and economics all fit within the decision-making framework [20]. The importance of the development of reliable methods to automatically solve sequential decision tasks cannot be overemphasized.

One of the main issues regarding the solution of sequential decision tasks is the so-called *temporal credit-assignment problem*: how to apportion credit to individual decisions by looking at the outcome of a sequence of them [16]. For example, after being surprised by a checkmate, a chess player would like to know which moves were responsible for the defeat and which were not. There are two main approaches to address the temporal credit-assignment problem:

– One may perform the credit assignment implicitly, by evaluating each decision policy as a whole and then combining the most successful ones in the

hope that better policies will come forth. In the example of chess, each decision policy would be a strategy to play the game. This type of *phylogenetic* learning is the approach adopted by evolutionary methods [5].

– Another possibility is to resort to an *ontogenetic* learning paradigm, in which a single decision policy is gradually refined based on the individual evaluation of the decisions made (in the game of chess each move would be evaluated separately). This is the basic idea behind reinforcement-learning algorithms [17].

The advantages and drawbacks associated with the phylogenetic and the ontogenetic learning have been the subject of some debate. Sutton and Barto [17] argue that evolutionary methods cannot be considered as true reinforcement-learning techniques because they are not able to use valuable information available during the learning process. In contrast, Moriarty *et al.* [12] list several advantages of using evolutionary methods to solve sequential decision problems. Indeed, many researchers have successfully applied evolutionary algorithms to decision tasks, not rarely obtaining better results than reinforcement-learning methods on the same tasks [22, 11, 6]. On the other hand, it is also possible to find reports of experiments in which evolutionary methods failed entirely while reinforcement learning performed well [1].

The aim of this work is to contribute to the ongoing discussion by noting that the performance of both evolutionary computation and reinforcement learning can be strongly influenced by the characteristics of the task at hand. Thus, instead of asking which approach is the best choice to solve sequential decision problems in general, one should try to identify the characteristics of a task that make it more or less amenable to be solved by each technique. Ideally, one would have well-defined categories of decision problems in which the expected performance of evolutionary computation and reinforcement learning was known. This way, the choices involved in the solution of a given problem would be less subject to intuition or personal inclination of the designer.

This paper represents a small step towards the scenario described above. It starts with a brief review of the criteria already used to classify decision problems and discusses the expected behavior of evolutionary computation and reinforcement learning on the resulting categories. This is done in Section 2. Then, in Section 3, a subdivision of one of these categories is proposed. As will be seen, this new dimension of classification is of particular interest for the evolutionary-computation community, since it clearly delineates a class of decision problems in which optimization methods are simply not applicable. The new categories also help to understand the disparity on the reports of previous experiments comparing reinforcement learning and evolutionary methods. Section 4 presents a simple computational experiment to illustrate the issues discussed in the previous section. Finally, the main conclusions regarding the present investigation are presented in Section 5.

2 Classification of Sequential Decision Problems

Sequential decision problems can be dealt with at different levels of abstraction. In the model considered here an *agent* must learn how to perform a task by directly interacting with it—thus, the task is sometimes referred to as the *environment* [17]. The interaction between agent and environment happens at discrete time intervals. At each time step t the agent occupies a state $s_i \in S$ and must choose an action a from a finite set A. The sets S and A are called the state and action spaces, respectively. The execution of action a in state s_i moves the agent to a new state s_j, where a new action must be selected, and so on.

Each transition $s_i \xrightarrow{a} s_j$ has an associated reward $r \in \mathbb{R}$, which provides evaluative feedback for the decision made. The use of rewards is a simple mechanism to represent the tradeoff between immediate and long-term benefits present in sequential decision problems. For example, if one wishes to model the game of chess, it suffices to associate a positive reward with every transition leading to a win and a negative reward with those transitions taking to a defeat. In this case, maximizing the expected reward corresponds to maximizing the probability of winning the game [17]. Given the above model of the decision-making process, a sequential decision problem can be classified as:

1. **Markovian / Non-Markovian:** In a Markovian decision problem the transitions and rewards depend only on the current state and the action selected by the agent [13]. Another way to put it is to say that Markovian states retain all the relevant information regarding the dynamics of a task: once the current state is known, the history of transitions that took the agent to that position is irrelevant for the purpose of decision making. In the game of chess, for example, a particular configuration of the board is all what is needed for an informed decision regarding the next move. On the other hand, upon deciding whether or not to concede a draw, a player may benefit from information about the history of previous games.

Reinforcement-learning algorithms were developed based on the Markovian assumption. In particular, they rely on the concept of a *value function*, which associates every state-action pair (s, a) with the expected reward following the execution of a in s [17]. It is not hard to see, therefore, that reinforcement learning is particularly sensitive to the Markov property: if the dynamics of a task depend on the history of transitions executed by the agent, it makes little or no sense to associate (s, a) with a specific sequence of rewards. This is not to say that it is impossible to apply reinforcement learning to non-Markovian tasks; however, at the current stage of theoretical development a non-Markovian problem still represents a considerable obstacle for reinforcement-learning algorithms in general. On the other hand, evolutionary algorithms do not associate credit with individual actions, but rather evaluate decision policies as a whole. Therefore, each state-action pair is considered in the context of an entire trajectory of the agent. This focus on the net effect of a sequence of actions is much more robust with respect to the Markov property (for a particularly enlightening example, see the article by Moritarty *et al.* [12]).

2. **Deterministic / Stochastic:** In a deterministic decision problem the execution of action a in state s_i always takes the agent to the same state s_j. In contrast, in a stochastic environment each transition is associated with a probability distribution over the state space S—that is, the agent may end up in different states on two distinct executions of a in s_i. The game of chess is clearly deterministic, whereas blackjack is an example of stochastic task [17].

A deterministic task is a degenerate stochastic environment in which the probability distributions associated with the transitions have only one nonzero element. Thus, any method designed for the latter category of decision problems should in principle also work in the former. Reinforcement-learning algorithms were developed with the stochastic scenario in mind, and apart from minor technical details the above distinction is irrelevant for them. Evolutionary computation can also be applied to both deterministic and stochastic tasks, but the latter require some caution. Since in stochastic tasks the same sequence of actions may result in different trajectories, an individual should not be evaluated on the basis of a single interaction with the environment (this would be like evaluating a chess player based on a single game). One way to circumvent this difficulty in evaluating candidate solutions is to resort to one of the many techniques available to deal with a noisy evaluation function (see the article by Moriarty *et al.* [12]).

3. **Small / Large:** Here, the terms "small" and "large" refer to the size of the state space with respect to the storage capacity of current computers. If a sequential decision problem is such that all state-action pairs can be stored in a look-up table, this problem is considered small. If S is large enough to preclude such storage, the task is considered large. Obviously, a decision problem with a continuous state space is always large.

As discussed above, standard reinforcement-learning algorithms associate every state-action pair with a number, which amounts to a storage requirement of $O(|S||A|)$. Obviously, when S is large this requirement cannot be fulfilled, and, therefore, one must resort to some form of approximation. Unfortunately, it is well known that the combination of reinforcement learning with general function approximators can easily become unstable [2, 18, 19]. One alternative is to use approximators with a linear dependence on the parameters, which results in stable reinforcement-learning algorithms [18, 9]. However, the performance of such algorithms is generally very sensitive to the set of features used in the approximation, which, in principle, must be handcrafted. Evolutionary algorithms can be easily combined with any type of approximator, and hence they can be more naturally applied to decision problems with a large state space.

4. **Stationary / Non-stationary:** In a stationary decision problem the dynamics of the environment are fixed, that is, the rules governing the transitions of the agent and the delivery of rewards do not change over time. A non-stationary environment is characterized by a changing environment, which means the performance of a decision policy may also vary with time.

A non-stationary decision problem is considerably more difficult than a stationary one for both reinforcement learning and evolutionary algorithms. Nevertheless, both approaches can be made to work in these problems with slight modifications in their standard forms. In the case of reinforcement learning, a non-stationary environment imposes a particularly severe version of the exploration/exploitation dilemma [17]. In order to address this problem, one must adopt techniques to guarantee a constant exploration of the environment, such as offering bonus rewards in states that have not been visited for a long time [17]. In the case of evolutionary computation, the standard strategy to address non-stationariness is to maintain the diversity within the population of candidate decision policies [12]. This way, changes in the dynamics of the problem will favor policies that perform better on the new environment, guiding the evolutionary search in the right direction.

5. **Episodic / Continual:** In an episodic task the interaction between agent and environment can be naturally broken into trials or *episodes* [17]. Each episode ends in a special state called a *terminal state*. There might be one or several such states. In chess, for example, every configuration of the board representing the end of a game would be a terminal state. In continual tasks the decision process goes on endlessly, without a clear criterion for interrupting it. One example of continual task is the automatic stock trading, in which an agent must decide whether to buy or sell stocks depending on the market situation.

Though not as commonly noted as the previous four categories, the distinction between episodic and continual tasks is of particular interest for evolutionary computation. It is well known that the search performed by evolutionary algorithms is based on information gathered between episodes, but not within them. This creates difficulties in the evaluation of candidate decision-policies for a continual task. Since in this case there is no clear criterion for interrupting the interaction of the agent with the environment, the evaluation of an individual must be truncated at an arbitrary point. Depending on the problem, it might be difficult to determine how much experience is necessary to properly measure the quality of an individual. From the reinforcement-learning perspective, the distinction between episodic and continual tasks is mostly irrelevant, since in the ontogenetic paradigm learning takes place both inter- and intra-episodes.

Notice that the dimensions of classification discussed above are orthogonal to each other, that is, the inclusion of a decision problem in one category does not influence its classification with respect to another criterion. This amounts to 2^5 different classes of decision problems. Obviously, it is possible to refine the above classification system by adding other dimensions to it. For example, at a higher level of generality it is possible to distinguish problems with a continuous action space from those with a finite number of actions available. Similarly, one can consider tasks in which the interaction between agent and environment happens continuously rather than at discrete time intervals [13]. The next section discusses another way to extend the above classification scheme.

3 A New Dimension of Classification

Among the classification criteria discussed in the previous section, the distinction between continual and episodic tasks is especially important for evolutionary methods. As discussed, continual tasks may create difficulties for these methods because the evaluation of an individual must be interrupted at a somewhat arbitrary point. Though this is certainly an obstacle, it does not prevent the use of evolutionary computation in the solution of continual decision-problems. In fact, this issue is similar to that of *generalization* in supervised learning, a well-understood problem for which several effective techniques exist [8].

Based on the discussion above, one may come to the following conclusion: evolutionary methods work well on episodic problems and, as long as the right techniques are employed, they can also be used on continual tasks. Unfortunately, the situation is a bit more complicated than that. Surprisingly enough, the most severe limitation of evolutionary computation manifests itself not on continual tasks, but on a particular type of episodic decision problem. As discussed before, in episodic tasks the interaction between agent and environment ends when the former reaches one of possibly many terminal states. But terminal states are not all the same; some represent desirable situations while others represent situations one would rather stay away from. This gives rise to the following definitions: if a terminal state is associated with the accomplishment of a task, such as winning a game or finding the exit of a maze, it is called a *goal state*. If, on the other hand, a terminal state represents an error of the agent, it is called a *dead-end*. Examples of dead-ends include: dropping the ball in a game like volleyball, falling off a suspended race track, losing all the money in a game of chance. This distinction between different types of terminal states induces a division of episodic tasks into two sub-categories:

5.1. Goal seeking / Error avoidance: An episodic decision problem is called a goal-seeking task if its terminal states represent the accomplishment of the task. Usually, the arrival at a goal state is accompanied by a positive reward or by the ceasing of a stream of negative rewards. In an error-avoidance episodic-task the terminal states are associated with undesirable situations. More specifically, the terminal states are "dead-ends" representing bad decisions that are irreversible. The delivery of rewards in an error-avoidance task follows the opposite logic of that of goal-seeking tasks.

Notice that, in contrast with the categories discussed in the previous section, the above classes of decision problems are not mutually exclusive, since it is possible for a task to simultaneously have goals and dead-ends. For example, Randløv and Alstrøm [14] proposed a task in which the objective is to balance and ride a bicycle to a target location. Thus, one must avoid terminal states representing the falling off the bike while seeking for those terminal states associated with the goal region.

In principle, there is no reason to believe the distinction between goal-seeking and error-avoidance tasks is of any relevance to reinforcement learning. In fact,

it is possible to find in the literature examples of reinforcement-learning applications which can be clearly identified with both categories [17]. In contrast, the performance of evolutionary methods can be highly influenced by the type of episodic decision-problem at hand. Usually, error-avoidance tasks cause no trouble for evolutionary computation. This is because in this type of problem it is straightforward to rank unsuccessful candidate solutions: the longer it takes for a decision policy to reach a dead-end, the better its evaluation. As a side effect, in an error-avoidance task the time spent in the evaluation of an individual is normally proportional to its quality. This is also a very desirable property, since it induces a smart allocation of computational resources: the evolutionary process will quickly evaluate a large number of poor solutions and eventually focus on candidate decision-policies that are worth the computational effort.

Unfortunately, the situation with goal-seeking tasks is quite different. In this type of task one is interested in finding a decision policy able to reach a specific set of states. Since the position of such states is not known beforehand, it is not clear how to compute the kind of quality measure required by the phylogenetic search, such as the distance from a given state to the closest goal. This makes it hard for evolutionary algorithms to differentiate between candidate solutions that are unable to accomplish the task. Of course, there are situations in which it is possible to estimate the quality of a decision policy based on additional information about the problem. However, when such information is not available, the evolutionary process is reduced to a random search until a successful solution luckily emerges in the population. Depending on the difficulty of the task, this might be a very unlikely event [1].

Notice that this question is not related to design choices such as the type of representation or genetic operators used by the evolutionary algorithm. In fact, this issue is intrinsic to any goal-seeking task and any optimization method which basis its search on the relative merit of candidate solutions. Reinforcement learning algorithms do not suffer from this difficulty because they evaluate each decision made by the agent individually. Hence, even before reaching the goal, the agent has an estimate of the effects of the decisions made so far.

The distinction between goal-seeking and error-avoidance episodic-tasks sheds some light on the apparent inconsistency on previous accounts of experiments comparing reinforcement learning and evolutionary algorithms. For example, in a sequence of scientific works initiated in 1993 by Whitley *et al.* [22], several researchers report an overwhelming advantage of evolutionary methods over reinforcement learning on experiments with a control problem known as the pole-balancing task [11, 10, 7, 15, 6]. In the pole-balancing task one has to apply forces to a wheeled cart moving along a limited track in order to keep a pole hinged to the cart from falling over. When formulated as a Markovian problem, the characteristic of pole-balancing that seems to favor evolutionary computation the most is the task's continuous state space. Nevertheless, the fact that it is an error-avoidance task may play an important role as well. The experiments of Barreto and Anderson [1] with the Acrobot task corroborates this hypothesis. The Acrobot is a continuous problem in which the objective is to help a

gymnast-robot swinging on a high bar to raise its feet above the bar. It is, therefore, a goal-seeking task. In their experiments with the Acrobot, Barreto and Anderson were unable to find a single decision policy able to accomplish the task when using an evolutionary algorithm. In contrast, reinforcement learning easily solved the problem.

4 An Illustrative Experiment

This section presents a computational experiment to illustrate the issues involved in the solution of episodic tasks using evolutionary methods. The experiment concerns a deterministic environment that can be easily configured as either an error-avoidance or a goal-seeking task.

In the proposed problem an agent must learn how to perform by directly interacting with a two-dimensional maze. The dynamics of the maze follow the convention usually adopted in this type of task: at every state, there are four actions available—`north`, `south`, `east` and `west`—whose effect is the movement of the agent in the corresponding direction. To allow for a broader analysis, the experiments were not performed on a single maze, but rather on a set of mazes with similar characteristics. All mazes were based on $n \times n$ grids and had one entrance at the upper-left corner. The position of the goal was selected at random. The mazes were generated in such a way that all states were connected to at least two other states (see Fig. 1).

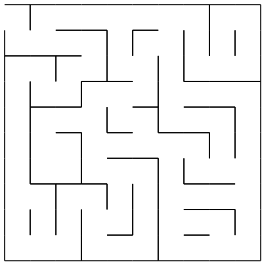

Fig. 1: Example of 10×10 maze

Two formulations of the problem were used. The first one is a typical error-avoidance task: the agent must travel inside the maze for as long as possible without hitting any walls (thus, the exit of the maze is ignored). If the agent runs into a wall, it gets a negative reward and is repositioned at the entrance of the maze. In order to avoid trivial solutions in which the agent repeatedly visits two neighbor cells, the state visited at time t was treated as a wall at time $t + 1$ (but not at $t + 2$). This forced the agent to look for cycles within the maze. The second version of the problem has the usual description of a maze: the agent must find a path from the entrance to the exit (that is, it gets a positive

reward when it finds the goal state). This is clearly a goal-seeking task. Notice that both versions of the problem are deterministic, stationary, and episodic. For the sizes of mazes used in the experiments, they are also small. Since in the error-avoidance task the transitions depend on the states previously visited by the agent, this is a non-Markovian problem. The goal-seeking task is Markovian.

The evolutionary method adopted in the experiments was a generational genetic-algorithm using a population of 100 individuals. The decision policies were represented as n^2-dimensional vectors, where n is the dimension of the maze. The elements of the vectors were one of the four actions available in the task, indicating the direction selected by the policy in each state of the maze. A two-point crossover was employed for recombination, with probability of occurrence of 0.9 [5]. After an individual had been created, the random mutation operator was independently applied to each of its elements with probability $1/n^2$. The candidate solutions were selected for reproduction based on linear ranking using a selective pressure of 1.5 [21]. At every generation the best solution found up to that point was copied to the next generation without modification.

As one would expect, the evaluation of candidate decision-policies was easy in the error-avoidance version of the task: the fitness of an individual was defined as the number of steps it performed before running into a wall. Notice that, considering the representation used, if a decision policy could avoid the walls for n^2 steps, it could do so indefinitely. Thus, upon performing this number of steps a decision policy was considered successful.

The definition of an evaluation function was not as straightforward in the goal-seeking version of the task. Since one does not know the number of steps separating a given state from the goal, it is necessary to resort to alternative sources of information to build an evaluation scheme that (hopefully) helps the agent to accomplish its objective. By analyzing the dynamics of the present task and the representation used by the genetic algorithm, it is clear that a decision policy cannot escape from the maze if it runs into a wall or visits a state more than once. Hence, the evaluation of an individual can be interrupted as soon as one of these events occur. Still, one is left with the problem of estimating the quality of a decision policy—that is, how far it is from completing the task.

In order to illustrate the issues involved in the derivation of an evaluation function for a goal-seeking task, two scenarios were considered in the experiments. In the first one the designer knows nothing about the task but its dynamics. In this case, a possible approach is to stimulate the candidate decision-policies to explore the maze as much as possible, in the hope that one will luckily find the goal. Thus, in this version of the task the fitness of an individual was defined as the number of steps it executed before hitting a wall or encountering an already-visited state. In the second formulation of the goal-seeking task the designer knew the coordinates of the goal state.[3] In this case, the fitness of an individual was defined as $n^2 - d$, where d is the Manhattan distance between the last state visited by the agent and the goal.

[3] The coordinates of the states were defined as their indices in the $n \times n$ grid underlying the maze.

Figure 2 shows the results obtained by the genetic algorithm after 500 generations on mazes of different sizes. Since the number of generations was fixed in the experiments, it is natural to observe a decrease on the success rate of the algorithm as the dimension of the mazes increases. However, the performance of the evolutionary method degenerates much faster on the goal-seeking task, as shown in the figure. This illustrates the issue discussed in the last section: since in the goal-seeking version of the problem the evaluation functions do not reflect the true objective of the agent, the performance of the algorithm deteriorates when a lucky move into the goal becomes less and less likely to occur.

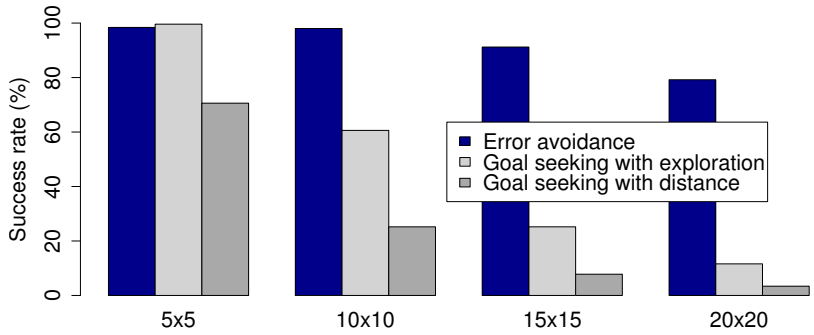

Fig. 2: Results obtained by the genetic algorithm on the maze task. In the error-avoidance task a run was considered successful if the agent could avoid the walls for n^2 steps. In the goal-seeking task a successful run was characterized by the agent finding the goal. The values correspond to an average computed over 500 mazes of each dimension. The same mazes were used for the goal-seeking and error-avoidance tasks.

Perhaps more surprising is the comparison between the two evaluation functions used in the goal-seeking task. Since the fitness based on the distance to the goal uses more information than its naive explorative counterpart, one would expect the former to generate better results. The reason why this is not so is unclear. The explanation might be related with the presence of multiple local minima represented by states close to the goal in the Euclidean sense but far away inside the maze. Incidentally, this experiment shows that the availability of information about a goal-seeking problem is no guarantee of success for an evolutionary method.

5 Conclusions

Evolutionary computation and reinforcement learning address the sequential decision-making problem in completely different ways, and their advantages and

drawbacks may be emphasized or disguised by the features of a specific task [17, 12]. Therefore, an important goal is to discover characteristics of a decision problem that can be easily identified and that at the same time provide some hint as to which of the two approaches should be adopted to solve the problem.

The contribution of this paper for the above scenario is twofold. First, it provides a systematic review of the criteria most commonly used to classify decision problems and discusses their impact on the performance of reinforcement learning and evolutionary computation. To the best of the authors' knowledge, this is the first attempt to organize this information in an easily accessible form. The paper also proposes the division of the class of episodic problems into two subcategories, which delimits a set of decision problems particularly difficult for optimization techniques in general and evolutionary methods in particular.

In an error-avoidance task the decision-making process lasts until the agent makes a mistake. This category of decision problem is usually amenable to evolutionary methods, since in this case it is trivial to evaluate decision policies that have failed to accomplish the task. On the other hand, in goal-seeking tasks unsuccessful candidate solutions cannot be easily ranked, since all decision policies unable to find a goal state are in principle equally bad. This subjects the use of evolutionary algorithms to the availability of prior information about the problem. There also exist episodic decision-problems whose characteristics can be identified with both error-avoidance and goal-seeking tasks. In problems with mixed features like this, evolutionary methods are expected to perform well without specific knowledge about the problem if the avoidance of errors helps the agent to get to the goal.

As discussed in the paper, the characteristics of a sequential decision problem have different effects over reinforcement learning and evolutionary algorithms: some characteristics will favor one over another, while others will have the same impact on both approaches. In some cases, the description of a given problem will make it obvious which paradigm one should resort to. In general, however, one should not expect the classification of a decision problem to provide such a clear-cut answer. Therefore, it might be a good idea to see evolutionary computation and reinforcement learning as complementary rather than mutually exclusive approaches. This seems to be the underlying assumption behind the *learning classifier systems*, a rule-based approach for solving decision problems which combines ideas from reinforcement learning and evolutionary computation [3]. It should be noted, however, that learning classifier systems still lack a strong mathematical basis, and much research must be done in order to understand the subtle interactions between the ontogenetic and the phylogenetic learning processes [4].

Acknowledgments

The authors would like to thank the support provided by the Brazilian agencies CNPq (grant 311651/2006-2), FAPERJ (grant E-26/102.825/2008), CAPES, and ANP.

References

1. A. M. S. Barreto and C. W. Anderson. Restricted gradient-descent algorithm for value-function approximation in reinforcement learning. *Artificial Intelligence*, 172(4-5):454–482, 2008.
2. J. A. Boyan and A. W. Moore. Generalization in reinforcement learning: Safely approximating the value function. In *Advances in Neural Information Processing Systems*, pages 369–376. MIT Press, 1995.
3. L. Bull and T. Kovacs, editors. *Foundations of Learning Classifier Systems*. Springer, 2005.
4. J. Drugowitsch. *Design and Analysis of Learning Classifier Systems—A Probabilistic Approach*. Springer, 2008.
5. D. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley Reading, MA, 1989.
6. F. Gomez, J. Schmidhuber, and R. Miikkulainen. Accelerated neural evolution through cooperatively coevolved synapses. *Journal of Machine Learning Research*, 9:937–965, 2008.
7. F. J. Gomez. *Robust non-linear control through neuroevolution*. PhD thesis, The University of Texas at Austin, 2003. Technical Report AI-TR-03-303.
8. T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2002.
9. M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.
10. D. E. Moriarty. *Symbiotic Evolution of Neural Networks in Sequential Decision Tasks*. PhD thesis, The University of Texas at Austin, 1997.
11. D. E. Moriarty and R. Miikkulainen. Efficient reinforcement learning through symbiotic evolution. *Machine Learning*, 22(1–3):11–32, 1996.
12. D. E. Moriarty, A. C. Schultz, and J. J. Grefenstette. Evolutionary algorithms for reinforcement learning. *Journal of Artif. Intelligence Research*, 11:241–276, 1999.
13. M. L. Puterman. *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.
14. J. Randløv and P. Alstrøm. Learning to drive a bicycle using reinforcement learning and shaping. In *Proc. of the Fifteenth Int. Conf. on Machine Learning*, pages 463–471, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.
15. K. O. Stanley. *Efficient evolution of neural networks through complexification*. PhD thesis, The University of Texas at Austin, 2004.
16. R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.
17. R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.
18. J. N. Tsitsiklis and B. V. Roy. Feature-based methods for large scale dynamic programming. *Machine Learning*, 22:59–94, 1996.
19. J. N. Tsitsiklis and B. V. Roy. An analysis of temporal-difference learning with function approximation. *IEEE Trans. on Automatic Control*, 42:674–690, 1997.
20. D. J. White. Real applications of Markov decision processes. *Interfaces*, 15:73–83, 1985.
21. D. Whitley. The GENITOR algorithm and selective pressure: why rank-based allocation of reproductive trials is best. In *Proc. of the Third Int. Conf. on Genetic Algorithms and their Applications*, pages 116–121. Morgan Kaufmann, 1989.
22. D. Whitley, S. Dominic, R. Das, and C. W. Anderson. Genetic reinforcement learning for neurocontrol problems. *Machine Learning*, 13(2-3):259–284, 1993.